A Method on Neural Network Optimization Deployment Frameworks for Lightweight Target Devices

Jaebok Park
AI Computing System SW
Research Section
ETRI
Daejeon, Korea
parkjb@etri.re.kr

Kyunghee Lee
AI Computing System SW
Research Section
ETRI
Daejeon, Korea
kyunghee@etri.re.kr

Jiyoung Kwak
AI Computing System SW
Research Section
ETRI
Daejeon, Korea
jiyoung@etri.re.kr

Changsik Cho
AI Computing System SW
Research Section
ETRI
Daejeon, Korea
cscho@etri.re.kr

***Abstract*—The development of artificial intelligence neural networks requires specialized knowledge in the industry. The process of creating and deploying neural networks is very difficult for software developers who lack knowledge. Therefore, there is a need for tools that can easily develop neural network applications in industries. In order to meet these needs, this paper proposes a deployment optimized for a target device by automatically generating a specification-based neural network. First, the user simply selects the desired system and neural network requirements. The proposed framework utilizes user specifications to generate the desired neural network. Next, template code is generated based on the generated neural network. Finally, the neural network is deployed to target devices. In this way, we present a method to build an artificial intelligence neural network application deployment easily and quickly.**

Keywords— Neural Network, AutoML, Deployment, NAS, Deep

I. INTRODUCTION

NAS(Neural Architecture Search)[1][2] can automate the design of neural networks by searching for the type and form of the most suitable neural network structure to solve a specific problem through deep learning. NAS is an automated technology adopted by data scientists to build and optimize neural networks for their intended purpose.

Even in small businesses, rapid development of neural network applications is required. For these needs, neural network deployment methods such as the searching of neural networks and the optimization of a target device are required. For fast deployment of neural network application services, we have to solve problems such as repeated experimentation of the optimal technique and the repeated error verification of neural networks.

The neural network application development process can be divided into neural network creation and neural network deployment. First, the neural network searches for a desired neural network in the neural network space according to the purpose. Then, the neural network is trained according to the purpose. This neural network creation process requires an expert level.

Neural network deployment requires optimized deployment in various HW acceleration environments to meet the performance requirements of the target. In addition, the deployment requires technical know-how in optimizing neural network acceleration in a cloud environment, an on-device environment with various acceleration environments, a distributed environment in which edge and edge devices are linked, and a collaborative environment in which various sensor devices are linked.

This paper presents an automatic neural network generation and deployment framework that can optimally deploy to a desired target device. The proposed framework searches a neural network according to user requirements, and the neural network is trained using the generated neural network and learning data. Finally, the desired neural network is optimized and provided. The created neural network automatically optimizes the neural network by considering the target devices performance according to the requirements. Based on the learned neural network and requirements, the final application template code is generated by considering the input and output of the neural network, the input of inference data, and the output of inference values. The generated application template code and the trained neural networks can be easily deployed on the target device through a container build.

The neural network deployment system is divided into a master device and a slave device according to user requirements, and generates a container-type neural network inference engine and deploys it to each device. Additionally, we also offer single device and single deployment on edge devices. The optimized deployment framework presented in this paper solves difficult problems from neural network searching, training, and target optimal deploying for the rapid development of neural network application services. In particular, a neural network is generated according to user requirements. We proposed a method for creating and deploying application template codes using the generated neural network.

II. PROPOSED AUTOMATIC NEURAL NETWORK GENERATION AND DEPLOYMENT FRAMEWORK

The automatic neural network creation and deployment tool makes it easy and fast to search for neural networks by simply filling out user specifications to develop and operate deep learning applications. It is a technology that automatically generates and distributes desired learning neural networks and application template codes. In particular, this automatic neural network distribution tool provides developers with insufficient neural network knowledge to easily and quickly develop and deploy desired neural network application services.

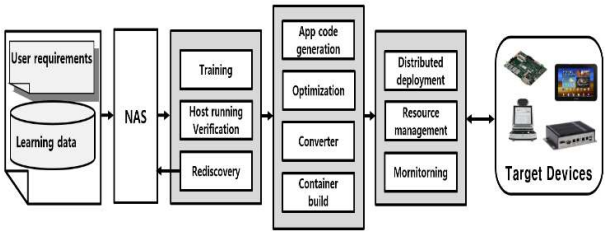

Fig. 1. The process of automatic neural network creation and deployment

As shown in Figure 1, the automatic neural network creation and deployment tool inputs requirements such as application, target device, performance, and training data for learning. Through neural network search technology, a neural network suitable for user requirements and learning data is generated. Basically, proposed framework retrieves the basic neural network architecture through NNI[3] and generates a Yolo-based[4] detection neural network structure. Automatic neural network generation is a target adaptive neural network search technology, which provides automatic generation of neural networks with simple input. In the project manager, a neural network is generated with only a few inputs, such as data set, target device, application service, and learning method. The execution process is displayed (AutoNN start point/Epoch/end point, etc.) in the console window using the color of the execution module after reorganization (distribution and loading).

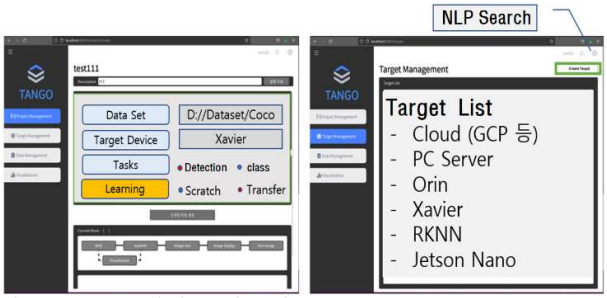

Fig. 2. Easy target device setting using GUI

As shown in Figure 2, when adding a device, NLP is provided to fill in the device performance figures. Multi-device selection gives you how to decide (Cloud, PC, Ondevice, etc.) In detail, the model selection module integrates the model selection algorithm into the automatic generation of specification-based and visualization-based neural networks. This neural network model expands the Yolov7 neural network model and the Resnet neural network source code, and generates a neural network Yaml suitable for each device. The automated machine learning pipeline of automatic neural network generation is shown in Figure 3 below.

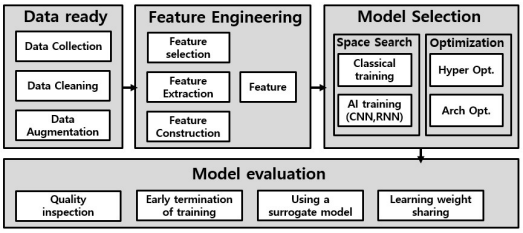

Fig. 3. Machine Learning Pipeline for Automatic Neural Network Creation

In the automatic neural network creation, hyper parameter optimization (HPO) optimizes hyper parameters to improve the performance of neural network models. In order to design the optimal hyperparameter search technology considering the data set and the neural network model, we analyzed the factors related to the performance of the model. As a result, if the LR (Learning Rate) is small, the learning rate is slow, but the performance error is small. Conversely, as the LR increases, the learning rate speeds up, but the performance error increases. Therefore, it was designed with a relatively large Cyclical Momentum applied to LR.

We train the neural network using the generated neural network and the training data entered by the user. Next, the application code is generated, and the neural network is optimized based on the requirements of the target. As shown in Figure 5, the operation flow receives the neural network information generated by the module of automatic neural network generation. The deployment information is described after generating the neural network execution code. Finally, through deploying, it is delivered to the trained neural network in the target device.

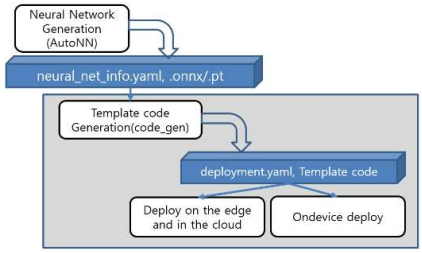

Fig. 5. Operation flow for deploying neural networks

III. EXPERIMENT AND VALIDATION OF THE GENERATED NEURAL NETWORK

In this section, we present a deployment method that optimizes accelerator-based adaptive execution codes. First, we optimized the pytorch-based[5] neural network file (.pt) generated by the automatic neural network generation module. The pt file generated by the automatic generation module was optimized from automatic generation to learning process. Therefore, in the deployment stage, the accelerator-based adaptive execution image is optimized and deployed. First, our optimization method proceeds with optimization by converting pt files to onnx[6]. The converted onnx-based neural network code is converted to fit the target. Currently, we are developing to consider the adaptive deployment of various target systems.

In this paper, Nvidia-based TensorRT[7] converting and Rockchip-based RKNN accelerated converting are provided. First, neural network technology can be developed using the TensorRT engine to improve the weight and speed of the target device. The pt file generated by automatic neural network creation matches the input and output of the neural network, and converts it to ONNX. The command below

converts the pt file into the onnx file, and the next command converts the converted onnx file into the Tensorrt-based code.

```
python3 export.py —weights yoloe.pt --grid --end2end --simplify --topk-all 100 --iou-thres 0.65 --conf-thres 0.35 --img-size 640 640
python3 convert.py –o yoloe.onnx -e yoloe.trt -p fp16
```

The converter creates a Tensorrt based neural network with nms and float point 16. Table 1 analyzes the performance using the our generated neural networks. We compared the performance of Jetsot AGX Orin[8] and Jetson AGX Xavier[9] using video files. As shown in the table, there is a performance difference between torch acceleration and Tensorrt acceleration. Through this, it was possible to confirm optimization from neural network creation to deployment.

TABLE I. PERFORMANCE ANALYSIS OF DEPLOYED NEURAL NETWORKS

	Jetson AGX Orin		**Jetson AGX Xavier**	
	Torch	***TensorRT***	***Torch***	***TensorRT***
yoloe-tiny	62FPS	229FPS	50FPS	184FPS
yoloe	39FPS	104FPS	32FPS	62FPS
yoloex	35FPS	68FPS	31FPS	44FPS

Next, the process of creating a Rockchip NPU neural network is as follows. First, in order to run a neural network on the NPU, a neural network model such as Caffe, TensorFlow, TensorFlow Lite, ONNX, or Darknet must be converted into an RKNN-based neural network. The converter is supported in the form of Docker in TANGO's[10] deployment module. TANGO is being developed using the proposed technologies. The neural network optimization supports quantization methods including an asymmetric quantization and a dynamic fixed point quantization. The RKNN's compilation uses pre-compilation technology to reduce model loading time. Pre-compiled RKNN models can only run on hardware platforms with NPUs. The RKNN-based neural network lightweighting proceeds through a conversion process as shown in Figure 6.

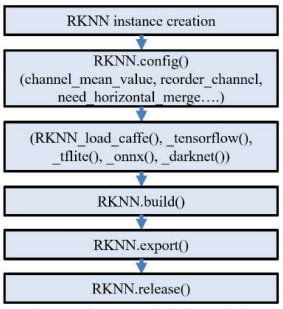

Fig. 6. RKNN-based neural network lightweight conversion process

For performance analysis, we analyzed using Odroad-M1[11] equipped with Rockchip. Odroid-M1 is equipped with Rockchip's RK3568B2. Specifications of the Rockchip RK3568B2 processor dedicated to the single-board computer sector, it has 4 cores and a maximum frequency of 2.0 GHz. We used Yoloe and existing yolo models for validation. The function was verified by operating the converted RKNN neural network along with the target template code in Odroid-M1.

The FPS of the RKNN-based neural network is shown in Table 2. The speed of the neural network is the highest in yolove-tiny. However, the accuracy may be low. However, it is very useful depending on the application in the industrial embedded board.

TABLE II. PERFORMANCE ANALYSIS OF RKNN-BASED YOLO NEURAL NETWORKS

Neural networks	***FPS(speed)***
yolov5n	13.6
yolov5s	6.4
yolov5m	4.3
yolov5l	2.4
yolov5x	1.4
yolove-tiny	14
yolove	2.4

IV. CONCLUSION

Developing artificial intelligence neural networks requires specialized knowledge. In particular, the process of creating and deploying neural networks is difficult for software developers. Therefore, a tool that can easily develop neural network applications is required. The proposed optimized deployment technology creates a neural network considering the environment of the target device. Next, an application template is provided so that the generated neural network can be optimized and easily operated in the target. We analyzed the usefulness of the generated neural networks using two targets. The analysis showed performance improvement of the optimization on Jetson series boards. RKNN-based analysis provided an easy and fast way to deploy neural networks on target. In the future, we will develop TANGO that can automatically deploy neural networks quickly and easily. In addition, various devices will be supported so that they can be used in various industries.

ACKNOWLEDGMENT

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No. 2021-0-00766, Development of Integrated Development Framework that supports Automatic Neural Network Generation and Deployment optimized for Runtime Environment)

REFERENCES

[1] M. Tan et al., “MnasNet: Platform-aware neural architecture search for mobile,” in Proc. IEEE Conf. Comput. Vision Pattern Recognit., 2019, pp. 2820.2828.

[2] Google, “Google AutoML Beta,” 2020. [Online]. Available: https://cloud.google.com/automl/, 2020.

[3] NNI(Neural Network Intelligence), https://github.com/microsoft/nni, 2022

[4] Glenn Jocher. Yolov5 in pytorch. https://github.com/ultralytics/yolov5, 2020

[5] Pytorch, https://pytorch.org/, 2022

[6] ONNX: Open Neural Network Exchange, https://onnx.ai/, Accessed in May, 2022

[7] NVIDIA Corporation, “Nvidia tensorrt,” [Online]. Available: https://developer.nvidia.com/tensorrt, 2023

[8] Nvidia Corporation, “Jetson Orin Technical Specifications,” [Online]. Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/, 2023

[9] Nvidia Corporation, “Jetson Xavier Technical Specifications,” 2023. [Online]. Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-agx-xavier/, 2023

[10] J. Park, H. Kim, S. Kim, K. Lee, C. Cho et al., https://github.com/ML-TANGO/TANGO, 2023

[11] Hardkernel Corporation, “Odroid-M1”, 2023